CS671: Deep Learning and Applications
Programming Assignment-1

Rohit Kumar
SCENE, IIT Mandi
D22078
D22078@students.iitmandi.ac.in

Nikhil Mahar
SCENE, IIT Mandi
D22123
D22123@students.iitmandi.ac.in

Sudhir Kumar Choudhary
SMME, IIT Mandi
(Project)

Abstract

Research in science and technology has came a long way from analysing a physical phenomena to mathematically modelling it and finding its solution. Large number of papers are available for finding analytical and numerical solution of mathematical models representing physical phenomena. However there are always some assumptions behind these models, which make them infeasible for actual problems. Therefore, instead of following Newtonian physics some of researchers followed the Galleolian theory of drawing inferences from large numbers of data. In early 90's some researchers tried to mimic functioning of brain and gave it a name of artificial neural network. Without any assumptions, they solved a problem just by learning from insights given by data. In this report we used single linear and non linear perceptron to classify three classes and also tried to approximate nonlinear univariate and bivariate solution.

1 Introduction

A single perceptron is the simplest form of a neural network, and it is also known as a linear threshold unit. It is a binary classifier that takes a set of inputs, multiplies them by a set of weights, adds a bias term, and applies an activation function to the resulting sum. The most used activation function for a perceptron is the step function, which returns 1 if the input is greater than or equal to a certain threshold value, and 0 otherwise. The threshold value is a parameter of the model that is learned during the training process. The weights and bias term of the perceptron are learned during the training process using an algorithm called the perceptron learning rule. The learning rule adjusts the weights and bias term of the model in response to misclassifications in the training data. The learning rule can be summarized as follows:

1. Initialize the weights and bias terms to small random values.
2. For each training example, compute the weighted sum of the inputs.
3. Apply the activation function to the weighted sum.
4. Update the weights and bias term if the predicted output is not equal to the true output.

5. Repeat steps 2-4 for a fixed number of epochs or until the model converges.

The single perceptron model is suitable for solving linearly separable problems, where a single straight line can separate the two classes in the input space. However, for non-linearly separable problems, a single perceptron is not sufficient, and more complex neural network architectures such as multi-layer perceptrons may be required. This report is broadly divided into two parts: (1) Classification, (2) Regression to measure capabilities of single linear and non-linear perceptron on linear and non-linear datasets.

2 Classification

In this section single non-linear perceptron model is used to classify the linearly separable and non-linearly separable datasets. A single neuron (perceptron) model with a sigmoidal activation function used for classification is shown below:

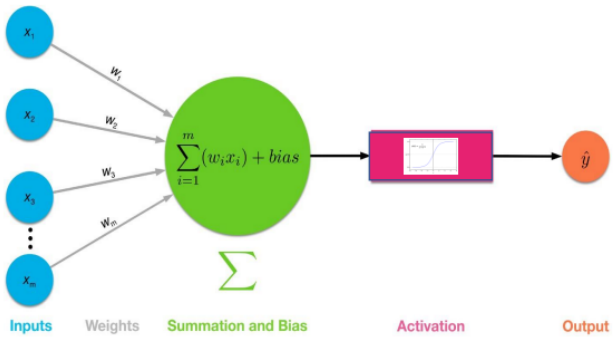


Figure 1: Schematic representation of perception model

2.1 Classification of linearly separable datasets

2.1.1 Dataset

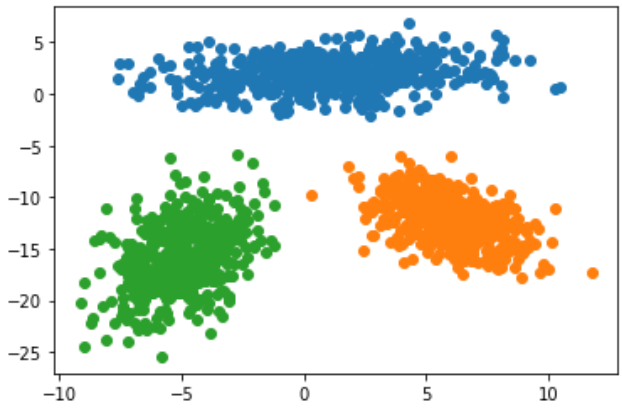


Figure 2: Datasets belonging to 3 classes, blue, orange, green as class 1, class2 and class3 respectively

2.1.2 Approach

For classification, 3 individual perceptron model are taken individually named as N12, N13, and N23. N12 is trained to distinguish class1 dataset from class2. N13 perception model is trained to distinguish between class 1 and 3. similarly N23 is trained for class 2 and class3. Finally majority voting scheme is used to plot all three decision boundaries together.

2.1.3 Results

1. Plot of average error vs epochs

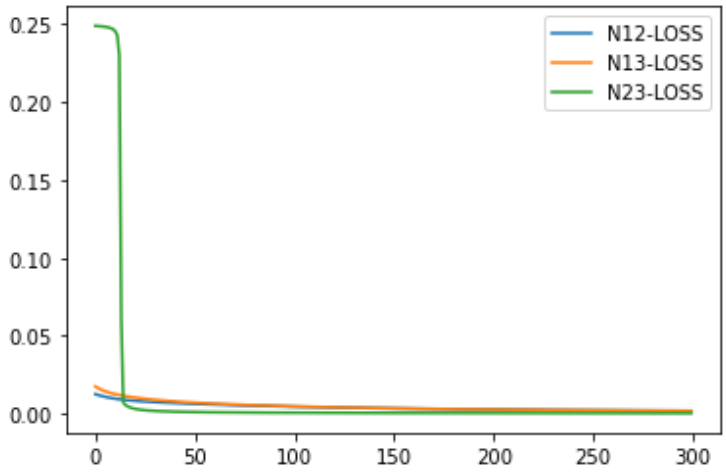


Figure 3: Datasets belonging to 3 classes, blue, orange, green as class 1, class2 and class3 respectively

2. Decision region plot superimposed by training data for each of the datasets.

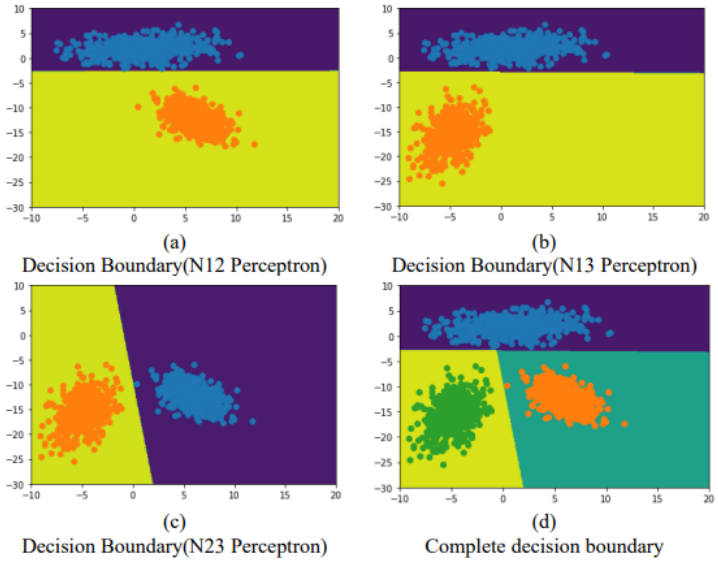


Figure 4: Decision boundary plots

3. Confusion matrix and classification accuracy.

- Confusion matrix for N12, N13, and N23 perceptron respectively,

N12	**Class 1**	**Class 2**
Class 1	150	0
Class 2	0	150

N13	**Class 1**	**Class 3**
Class 1	150	0
Class 3	0	150

N23	**Class 2**	**Class 3**
Class 2	150	0
Class 3	0	150

	Class 1	**Class 2**	**Class 3**
Class 1	300	0	0
Class 2	0	300	0
Class 3	0	0	150

- Accuracy:
  C11: 300 test samples predicted as class 1 actually belongs to class 1
  C12: 0 test samples predicted as class 1 actually belongs to class 2
  C13: 0 test samples predicted as class 1 actually belongs to class 3
  C21: 0 test samples predicted as class 2 actually belongs to class 1
  C22: 300 test samples predicted as class 2 actually belongs to class 2
  C23: 0 test samples predicted as class 2 actually belongs to class 3
  C31: 0 test samples predicted as class 3 actually belongs to class 1
  C32: 0 test samples predicted as class 3 actually belongs to class 2
  C33: 300 test samples predicted as class 3 actually belongs to class 3

$$Accuracy = \frac{(C11 + C22 + C33)}{(C11 + C12 + C13 + C21 + C22 + C23 + C31 + C32 + C33)} * 100 \quad (1)$$

  Accuracy=100%

2.1.4 Inferences

It can be observed from the above result section that single non-linear perceptron successfully classfied the decision boundaries of 3 datasets.

2.2 Classification of non-linearly separable datasets

2.2.1 Dataset

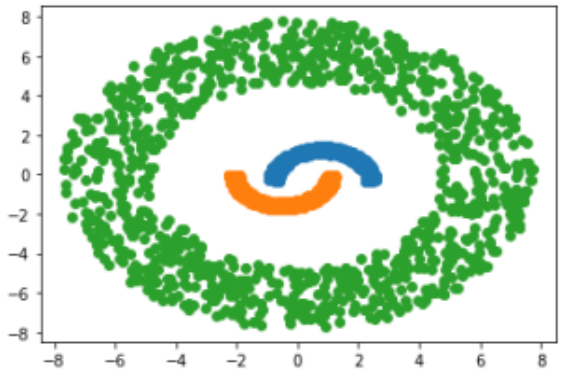


Figure 5: Datasets belonging to 3 classes, blue, orange, green as class 1, class2 and class3 respectively

2.2.2 Approach

For classification, 3 individual perceptron model are taken individually named as N12, N13, and N23. N12 is trained to distinguish class1 dataset from class2. N13 perception model is trained to distinguish between class 1 and 3. simillarly N23 is trained for class 2 and class3. Finally majority voting scheme is used to plot all three decision boundaries together.

2.2.3 Results

1. Plot of average error vs epochs

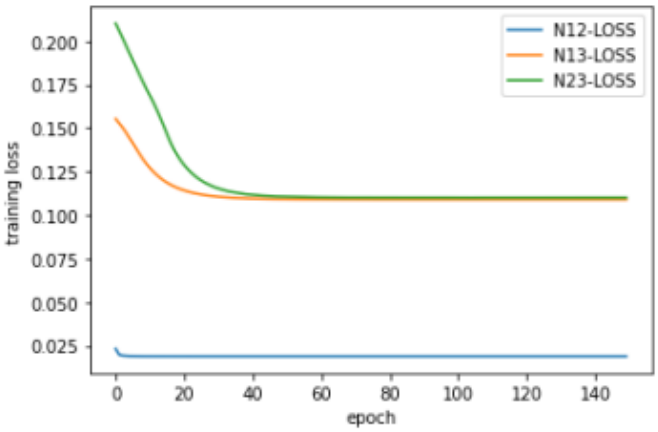


Figure 6: Datasets belonging to 3 classes, blue, orange, green as class 1, class2 and class3 respectively

2. Decision region plot superimposed by training data for each of the datasets.
3. Confusion matrix and classification accuracy.
    - Confusion matrix for N12, N13, and N23 perceptron respectively,

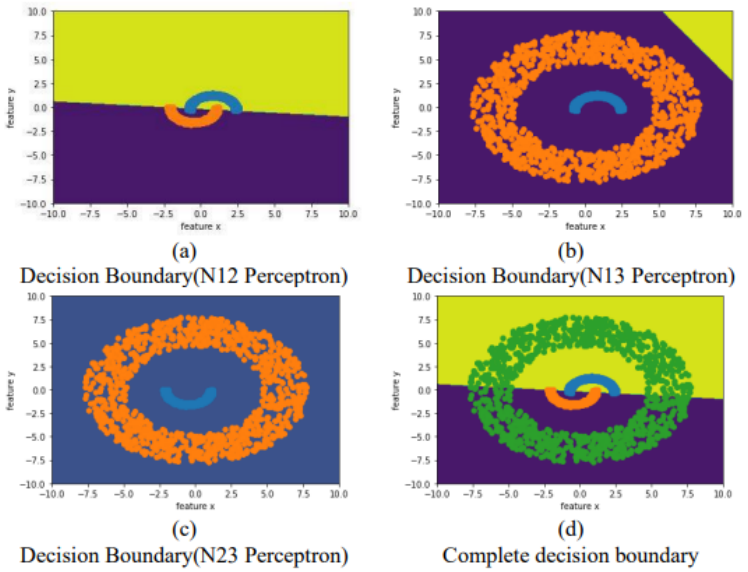


Figure 7: Decision boundary plots

N12	**Class 1**	**Class 2**
Class 1	146	4
Class 2	4	146

N13	**Class 1**	**Class 3**
Class 1	300	0
Class 3	150	0

N23	**Class 2**	**Class 3**
Class 2	300	0
Class 3	150	0

	Class 1	**Class 2**	**Class 3**
Class 1	446	0	0
Class 2	4	446	0
Class 3	150	150	0

- Accuracy:
  C11: 446 test samples predicted as class 1 actually belongs to class 1
  C12: 4 test samples predicted as class 1 actually belongs to class 2
  C13: 0 test samples predicted as class 1 actually belongs to class 3
  C21: 4 test samples predicted as class 2 actually belongs to class 1
  C22: 446 test samples predicted as class 2 actually belongs to class 2
  C23: 0 test samples predicted as class 2 actually belongs to class 3
  C31: 150 test samples predicted as class 3 actually belongs to class 1
  C32: 150 test samples predicted as class 3 actually belongs to class 2
  C33: 0 test samples predicted as class 3 actually belongs to class 3

$$Accuracy = \frac{(C11 + C22 + C33)}{(C11 + C12 + C13 + C21 + C22 + C23 + C31 + C32 + C33)} * 100 \quad (2)$$

Accuracy=74.33%

2.2.4 Inferences

As the dataset is highly non-linear, it can be observed from the above result section that a single non-linear perceptron failed to classify the decision boundaries of 3 datasets.

3 Regression

In this section single linear perceptron model is trained to fit the one dimensional univariate and bivariate data. A single neuron (perceptron) model with a linear activation function used for regression task is shown below:

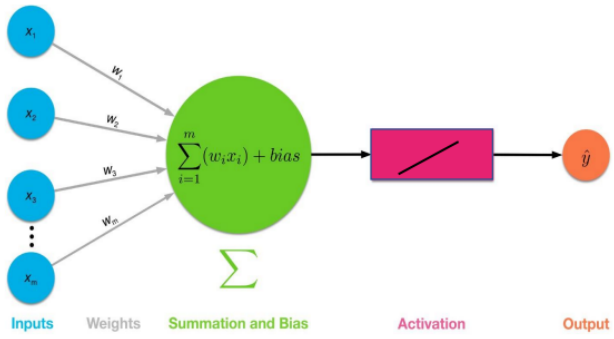


Figure 8: Schematic representation of perception model

3.1 Regression of univariate dataset

3.1.1 Dataset

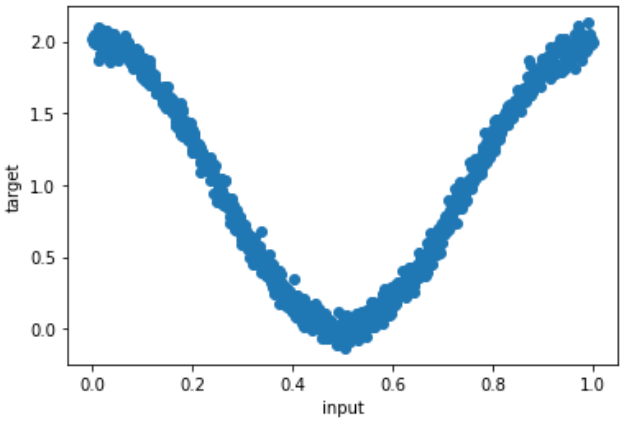


Figure 9: Univariate data required for training

3.1.2 Approach

For fitting the line around bivariate data, a single linear perceptron model is trained using univariate data as input. Input layer contained 1 neurons, while hidden layer contain 1 neuron. Network is trained with 70% of input data using stochastic gradient descent algorithm and tested on remaining 30% data. At each epoch different sequence of data is given as input so that model do not overfit the results.

3.1.3 Results

1. Plot of average error vs epochs and mean squared error of training and testing data.

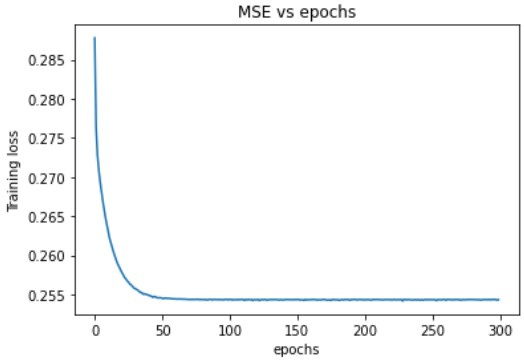


Figure 10: Average error vs epochs

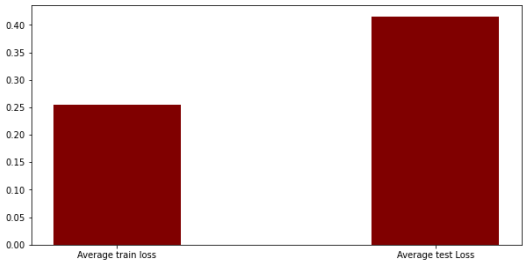
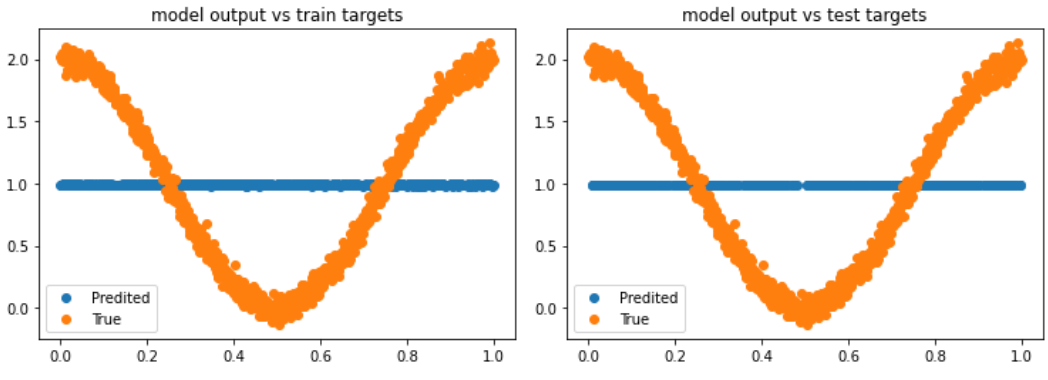


Figure 11: MSE of training and testing samples

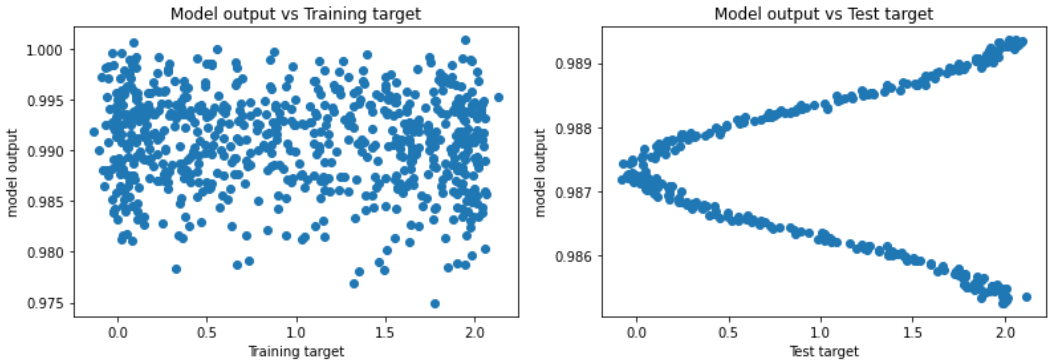
2. Plots of model output and target output for training data and test data



(a) (b)

Figure 12: (a) Model output vs train targets, (b) Model output vs test targets

3. Plots of model output and target output for training data and test data



(a) (b)

Figure 13: Scatter plot showing goodness of fit for (a) training, (b) test dataset

3.1.4 Conclusion

It can be observed from Figure 10 and Figure 11 that losses are very high and do not converge after certain number of epochs. As the data is non-linear our linear perceptron model fails to provide accurate results depicted by goodness of fit in Figure 13.

3.2 Regression of bivariate dataset

3.2.1 Dataset

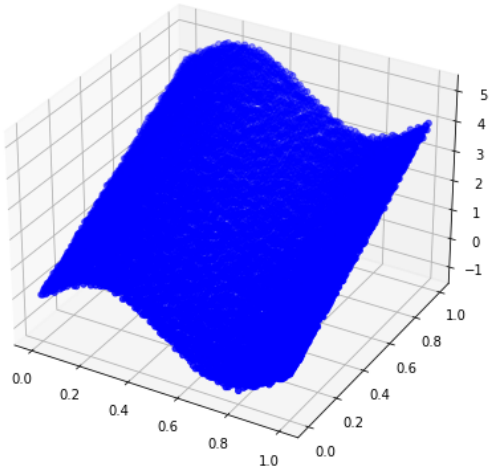


Figure 14: Bivariate data required for training

3.2.2 Approach

A single linear perceptron model is trained using bivariate data as input to fit the surface around bivariate data. Input layer contained two neurons, while hidden layer contain 1 neuron. Network is trained with 70% of input data using stochastic gradient descent algorithm and tested on remaining 30% data. At each epoch different sequence of data is given as input so that model do not overfit the results.

3.2.3 Results

1. Plot of average error vs epochs and mean squared error of training and testing data.

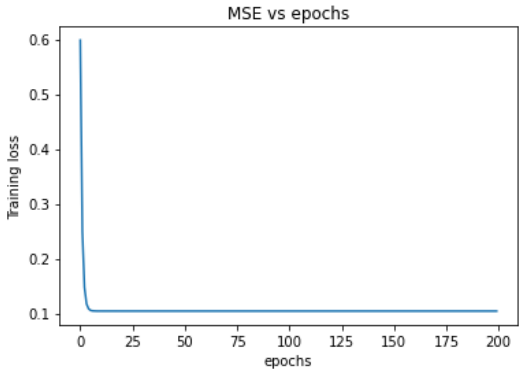


Figure 15: Average error vs epochs

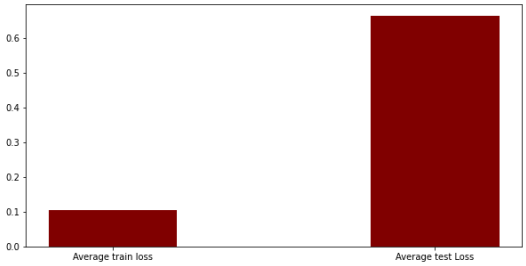
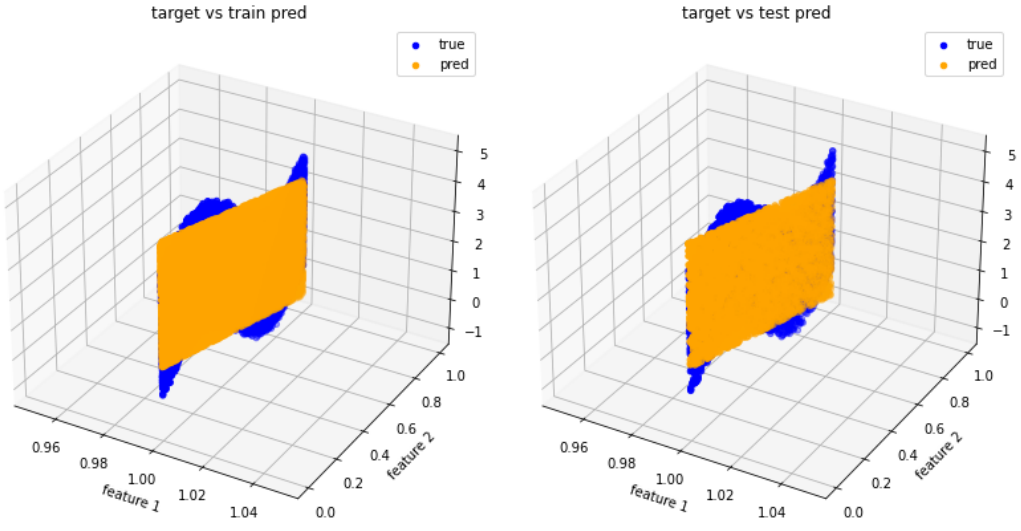


Figure 16: MSE of training and testing samples

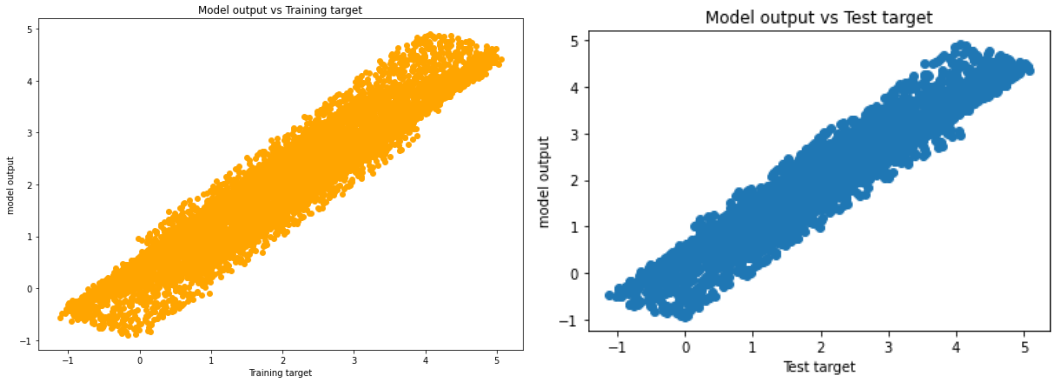
2. Plots of model output and target output for training data and test data



(a) (b)

Figure 17: (a) Model output vs train targets, (b) Model output vs test targets

3. Plots of model output and target output for training data and test data



(a) (b)

Figure 18: Scatter plot showing goodness of fit for (a) training, (b) test dataset

3.2.4 Conclusion

It can be observed from Figure 15 and Figure 16 that losses are very high and do not converge after certain number of epochs. As the data is non-linear, our linear perceptron model fails to provide accurate results depicted by the goodness of fit in Figure 18.

4 Data and codes

All the data and codes are provided in github repository(`https:https://github.com/nikhilmahar/CSE671_Asignment_1/new/main?readme=1`).

5 Contribution

Nikhil Mahar, Rohit Kumar and Sudhir Kumar have contributed equally.